
CLAIMS

[Claim(s)]

[Claim 1] A program counter and a register required to process a fixed number of program styles simultaneously, It is a multithread computer equipped with the controlling mechanism which performs pipeline control. A means by which this controlling mechanism generates and performs a new thread according to the 1st instruction code which specifies with an operand the start address of the task which can be performed to juxtaposition, the 2nd instruction code which specifies the task termination which can be performed to this juxtaposition -- responding -- this -- the multithread computer possessing a means to extinguish a new thread.

[Claim 2] The multithread computer which is a multithread computer equipped with a program counter and a register required to process a fixed number of program styles simultaneously, and the controlling mechanism which performs pipeline control, and possesses a means generate and perform two or more new threads according to the predetermined instruction code which specifies with an operand this start address of two or more tasks and the number of these tasks with

which this controlling mechanism has the same start address, and which can be performed to juxtaposition.

[Claim 3] Said controlling mechanism is a multithread computer according to claim 1 or 2 which has the thread ID register which stores the unique ID number added to the generated thread, respectively, and possesses further the means which reads the value stored in the thread ID register corresponding to the thread to which this instruction code belongs according to the instruction code which specifies this thread ID register.

[Claim 4] Said controlling mechanism is a multithread computer according to claim 1 or 2 which inhibits generation of the new thread according to instruction code, and possesses further a thread generation suppression means to perform an instruction sequentially according to the situation of a computer.

[Claim 5] Said controlling mechanism is a multithread computer according to claim 1 or 2 which possesses further the number control means of threads which controls the number of the threads by which a concurrency should be carried out among the generated new threads.

DETAILED DESCRIPTION

[Detailed Description of the Invention]

[0001]

[Field of the Invention] This invention relates to the multithread calculating machine which aimed at improvement in the operation effectiveness of a hardware resource in the detail more about a multithread calculating machine.

[0002]

[Description of the Prior Art] The multithread method is proposed and realized as a means which raises the activation throughput of a pipeline calculating machine. This is that that a single program style fully uses up the operation resource of the juxtaposition which a pipeline computer offers generally supplies two or more program styles to a pipeline simultaneously paying attention to a difficult point, and aims at reducing the so-called rates of an empty slot substantially.

[0003] On the other hand, the parallelism which exists in a program with a parallelization compiler is extracted, and approach which carries out parallel execution by the multiprocessor is performed from the former.

[0004]

[Problem(s) to be Solved by the Invention] By the way, although the thing of the magnitude of the part which can be processed to juxtaposition at once is called the grain size (granularity) about parallelism, if it is going to carry out parallel processing with a small grain size, an overhead can become large and cannot not much raise effectiveness. Conventionally, the multithread computer does not support the parallelism of a big grain size which a program has for the purpose of the parallel processing in a small grain size. The extract of the parallelism according to a parallelization compiler on the other hand

is performed on the level of a big grain size, and it cannot necessarily be said that the hardware resource is fully utilizing on fine level.

[0005] In view of the trouble which mentioned this invention above, the object is in offering the multithread computer which carries out multithread activation of the two or more programs style which was extracted with the parallelization compiler, and in which parallel execution is possible by the single processor aiming at fusion of two techniques mentioned above.

[0006]

[Means for Solving the Problem] A program counter and a register required to process a fixed number of program styles simultaneously according to the 1st field of this invention in order to attain the above-mentioned object, It is a multithread computer equipped with the controlling mechanism which performs pipeline control. A means by which this controlling mechanism generates and performs a new thread according to the 1st instruction code which specifies with an operand the start address of the task which can be performed to juxtaposition, the 2nd instruction code which specifies the task termination which can be performed to this juxtaposition -- responding -- this -- the multithread computer possessing a means to extinguish a new thread is offered.

[0007] Moreover, a program counter and a register required to process a fixed number of program styles simultaneously according to the 2nd field of this invention, It is a multithread computer equipped with the controlling mechanism which performs pipeline control. The multithread computer possessing a means by which this controlling mechanism generates and performs two or more new threads according to the predetermined instruction code which specifies with an operand this start address of two or more tasks and the number of these tasks which

can be performed to the juxtaposition which has the same start address is offered.

[0008] Moreover, according to this invention, the controlling mechanism has the thread ID register which stores the unique ID number added to the generated thread, respectively, and the means which reads the value stored in the thread ID register corresponding to the thread to which this instruction code belongs according to the instruction code which specifies this thread ID register is provided further.

[0009] Moreover, according to this invention, according to the situation of a computer, the controlling mechanism inhibits generation of the new thread according to instruction code, and possesses further a thread generation suppression means to perform an instruction sequentially.

[0010] Moreover, according to this invention, the controlling mechanism possesses further the number control means of threads which controls the number of the threads by which a concurrency should be carried out among the generated new threads.

[0011]

[Embodiment of the Invention] Hereafter, the operation gestalt of this invention is explained with reference to an accompanying drawing.

[0012] Drawing 1 is a block diagram for explaining actuation of a common multithread calculating machine. For an instruction cache and 14a-14d, as for a pipeline and 18a-18d, in this drawing, a program counter and 16 are [12 / a register file and 20] controlling mechanisms, respectively. Two or more program counters and register files exist, and the number is the upper limit of the number of threads which a processor (computer) supports in hardware and in which a concurrency is possible.

[0013] The fetch of each thread under activation is carried out from an instruction cache 12 using the address which the program counter to which it corresponds in 14a-14d shows, and it is supplied to a pipeline

16. The register file accessed into an instruction execution is chosen from either of the 18a-18d, and is used. The structure of such a multithread processor is well-known, and is premised on it also for this invention.

[0014] Drawing 2 is drawing showing the example of the program structure extracted by the parallelization compiler. In the example of this drawing, after that, Task A is performed, and control moves to two or more tasks B1 in which parallel execution is possible - B4, and the last joins one and serves as Task C at the beginning.

[0015] In 1 operation gestalt of this invention, it is putting a thread_fork instruction on the last of the task A of drawing 2 , and the framework which carries out multithread activation of a task B1 - B4 is offered. This thread_fork instruction has the task initiation address as an operand, as shown in drawing 3 .

[0016] If this thread_fork instruction is executed, the program style which makes a head the specified address will be defined as a new thread, and multithread activation will be carried out according to a well-known activation method. In the example of drawing 3 corresponding to drawing 2 , four thread_fork instructions corresponding to a task B1 - B4 will be put on the last of Task A by the parallelization compiler.

[0017] Moreover, the thread_terminate instruction showing the end of a thread is also prepared, and if this instruction is executed, a thread will disappear according to a well-known method. That is, by the parallelization compiler, as shown in drawing 4 , a thread_terminate instruction will be put on the last of a task B1. It is the same also about task B-2 - B4.

[0018] When the structure of drawing 2 is special, there is [******] an example of drawing 5 . Although drawing 5 is the same as drawing 2 as

flow of a program, the program has taken loop structure and the point that a task B1 - B4 are located in the same part in instruction code is special. In 1 operation gestalt of this invention, a multi_thread_fork instruction is prepared corresponding to this example.

[0019] This multi_thread_fork instruction has as an operand a thread starting address and the number of the threads which should be generated. That is, in the case of the example of drawing 5 , by the parallelization compiler, as shown in drawing 6 , a multi_thread_fork instruction is put on the last of the instruction train of Task A, and the starting address and four threads of a task B1 are specified as the operand. In this way, it becomes possible to carry out multithread activation of the task B1 of drawing 5 - B4.

[0020] In addition, in drawing 5 , Variable I serves as a value of " 2", when it is used as an index variable meaning the count of a repeat of a loop formation, for example, it is referred to by the task B1 and it is referred to by the value of " 1", and task B- 2. That is, when these tasks are performed by juxtaposition, reading appearance of the value which is different even if it is the same variable needs to be carried out, and a certain management is needed. In 1 operation gestalt of this invention, this problem is solved with a thread ID register.

[0021] Drawing 7 shows actuation when this thread ID register is specified as a source operand. In this drawing, for a thread control section and 24a- 24d, as for instruction code and 28, a thread ID register and 26 are [22 / a decoder and 30] selectors, respectively, and these elements are contained in the controlling mechanism 20 of drawing 1 .

[0022] Generally the multithread processor has the thread control section 22, and it equips the interior with the register groups 24a- 24d which store two or more threads ID under parallel execution.

[0023] As shown in drawing 7, by the signal 40 which a decoder 28

outputs, one register is chosen from the thread ID register groups 24a-24d, reading appearance of the value stored in it is carried out, and it is always outputted as a signal 42. It is possible to obtain the thread ID of the instruction which the decoder 28 has decoded according to this structure as a value of a signal 42.

[0024] A selector 30 chooses from two or more source operand candidates the suitable thing specified by instruction code, and it opts for the selection with the value of a select signal 44. There is a thread ID which is given by the signal 42 as alternative in addition to the usual source operand group 46, and the value chosen from them is outputted as a signal 48.

[0025] If the instruction code 26 which has a thread ID register in a source operand is given, a decoder 28 will interpret it and a value as which a selector 30 chooses a signal 42 will be set up as a select signal 44. By this, ID of the thread to which an instruction belongs can be read as a source operand value.

[0026] Thus, if a thread ID register is specified as an operand of an instruction, since reading appearance of the unique number (thread ID) will be carried out to each thread, a right value can be referred to, if Variable I (drawing 5) is accessed using this as shown in drawing 8. In addition, the content of the register r0 with which <r0> gives a base address, and <ID> show the content of the thread ID register among drawing 8 , respectively.

[0027] By the way, although the task which can be performed to juxtaposition was extracted by the compiler in the program of drawing 2 or drawing 5 , originally it is a sequential program, and even if it does not carry out parallel execution, it operates correctly. On the other hand, as for a multithread processor, it is common that the optimal number of concurrency threads changes with situations of operation, and in order

to raise the effectiveness ability of the whole system, it is indispensable [a processor] to carry out supervisory control of the number of threads appropriately.

[0028] In view of this point, the framework controlled that the number of threads which carries out parallel execution should be restricted is offered by this invention. Even if a thread_fork instruction is executed, one of them is made not to perform thread generation, when hardware (namely, controlling mechanism 20) judges the advantages and disadvantages of newly carrying out thread generation and judges automatically that loss is large from the operation situation of a processor (computer). As mentioned above, since it operates correctly even if it does not carry out parallel execution of the program itself, even if it performs such control, it is satisfactory.

[0029] Moreover, hardware does not make this judgment automatically but another gives the means which software, such as an operating system, controls. for example, as shown in drawing 9 , the thread generation suppression register in which setting out by software is possible should be prepared, and an operating system should control thread generation -- ** -- when it judges, the same effectiveness can be acquired by setting this register to ON.

[0030] According to this invention, generally, two or more threads will be generated simultaneously. In the example of drawing 2 or drawing 5, the number is four pieces. However, the number of threads from which the greatest effectiveness is acquired changes with concurrencies as mentioned above depending on the operation situation of the property of a program, or the whole system. Then, a means to determine whether the concurrency of how many [in the thread group generated simultaneously] should be carried out is prepared.

[0031] Drawing 10 is drawing explaining the device in which hardware

determines automatically the number of threads which carries out a concurrency, and controls it according to a situation. For a thread control section and 12, as for the instruction-execution section and 52, in this drawing, an instruction cache and 50 are [22 / the monitor section and 54] the number assignment registers of threads.

[0032] In an instruction cache 12, two or more instructions of the thread which can be performed exist, and parallel execution of a number of the threads specified with the number assignment register 54 of threads from the inside is chosen and carried out. Carrying out monitoring of the situations (operation resource usage etc.) of the instruction-execution section 50 suitably, the monitor section 52 determines the suitable number of threads, and sets the number of threads as the number assignment register 54 of threads through a signal 56.

[0033] Drawing 11 is drawing showing the configuration whose software enabled it to perform setting out of the number assignment register 54 of threads in drawing 10 . The signal 58 was being added to that the configuration shown in drawing 11 is indicated to be to drawing 10 . When analyses, such as a compiler, show the suitable number of parallel execution threads, it is not based on a hardware function like drawing 10 , but the number assignment register 54 of threads is set up through a signal 58 with software.

[0034] In the above, this invention was explained to the detail especially with reference to the gestalt of the desirable operation. The concrete gestalt of this invention is appended to below for an easy understanding of this invention.

[0035] (Additional remark 1) A program counter and a register required to process a fixed number of program styles simultaneously, It is a multithread computer equipped with the controlling mechanism which performs pipeline control. A means by which this controlling mechanism

generates and performs a new thread according to the 1st instruction code which specifies with an operand the start address of the task which can be performed to juxtaposition, the 2nd instruction code which specifies the task termination which can be performed to this juxtaposition -- responding -- this -- the multithread computer possessing a means to extinguish a new thread.

[0036] (Additional remark 2) Multithread computer which is a multithread computer equipped with a program counter and a register required to process a fixed number of program styles simultaneously, and the controlling mechanism which performs pipeline control, and possesses a means generate and perform two or more new threads according to the predetermined instruction code which specifies with an operand this start address of two or more tasks and the number of these tasks with which this controlling mechanism has the same start address, and which can be performed to juxtaposition.

[0037] (Additional remark 3) Said controlling mechanism is a multithread computer given in the additional remark 1 or additional remark 2 which has the thread ID register which stores the unique ID number added to the generated thread, respectively, and possesses further the means which reads the value stored in the thread ID register corresponding to the thread to which this instruction code belongs according to the instruction code which specifies this thread ID register.

[0038] (Additional remark 4) Said controlling mechanism is a multithread computer given in the additional remark 1 or additional remark 2 which inhibits generation of the new thread according to instruction code, and possesses further a thread generation suppression means to perform an instruction sequentially according to the situation of a computer.

[0039] (Additional remark 5) Said thread generation suppression means is a multithread computer given in the additional remark 4 which is what

performs an instruction sequentially as a result of having supervised the situation of a computer automatically.

[0040] (Additional remark 6) Said thread generation suppression means is a multithread computer given in the additional remark 4 which is what performs an instruction sequentially in response to directions of the software which supervises the situation of a computer.

[0041] (Additional remark 7) Said controlling mechanism is a multithread computer given in the additional remark 1 or additional remark 2 which possesses further the number control means of threads which controls the number of the threads by which a concurrency should be carried out among the generated new threads.

[0042] (Additional remark 8) Said number control means of threads is a multithread computer given in the additional remark 7 which is what controls the number of the threads by which a concurrency should be carried out as a result of having supervised the situation of a computer automatically.

[0043] (Additional remark 9) Said number control means of threads is a multithread computer given in the additional remark 7 which is what controls the number of the threads by which a concurrency should be carried out in response to directions of the software which supervises the situation of a computer.

[0044]

[Effect of the Invention] According to this invention, the multithread computer which carries out multithread activation of the two or more programs style which was extracted with the parallelization compiler, and in which parallel execution is possible by the single processor is offered, and the operation effectiveness of a hardware resource in which pipeline control is performed is made to improve further, as explained above.

[Translation done.]

PATENT ABSTRACTS OF JAPAN

(11)Publication number : **2003-167748**
(43)Date of publication of application : **13.06.2003**

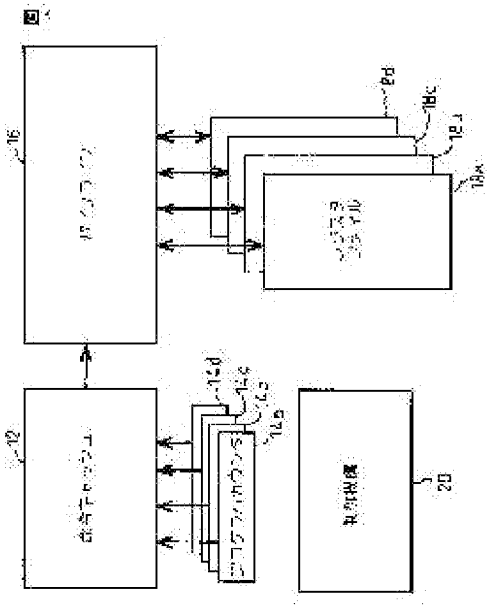
(51)Int.Cl. **G06F 9/46**
G06F 9/32
G06F 9/38

(21)Application number : **2001-366320**
(22)Date of filing : **30.11.2001**

(71)Applicant : **FUJITSU LTD**
(72)Inventor : **YASUSATO AKIRA**

(54) MULTITHREADED COMPUTER

(57)Abstract:
PROBLEM TO BE SOLVED: To provide a multithreaded computer that executes a plurality of parallel executable program streams extracted by a parallel compiler by means of a single processor in a multithreaded manner.
SOLUTION: The multithreaded computer comprises program counters and registers necessary for concurrent processing of a constant number of program streams, and a control mechanism for pipeline control. The control mechanism comprises a means for generating and executing a new thread according to a first instruction code designating a start address of a parallel executable task in an operand, and a means for erasing the new thread according to a second instruction code designating an end of the parallel executable task.



LEGAL STATUS

[Date of request for examination]

[Date of sending the examiner's decision of rejection]

[Kind of final disposal of application other than the examiner's decision of rejection or application converted registration]

[Date of final disposal for application]

[Patent number]

[Date of registration]

[Number of appeal against examiner's decision of rejection]

[Date of requesting appeal against examiner's decision of rejection]

[Date of extinction of right]

(19)日本国特許庁（ＪＰ）　(12) **公開特許公報（Ａ）**　(11)特許出願公開番号
特開2003－167748
（P2003－167748A）
(43)公開日　平成15年６月13日(2003.6.13)

(51)Int.Cl.[7]		識別記号	FI		テーマコード*（参考）
Ｇ０６Ｆ	9/46	３１３	Ｇ０６Ｆ　9/46	３１３Ｃ	５Ｂ０１３
				３１３Ｅ	５Ｂ０３３
		３４０		３４０Ｂ	５Ｂ０９８
	9/32	３１０	9/32	３１０Ａ	
	9/38	３７０	9/38	３７０Ｘ	

審査請求　未請求　請求項の数５　ＯＬ　（全　８　頁）

(21)出願番号　特願2001－366320(P2001－366320)
(22)出願日　平成13年11月30日(2001.11.30)

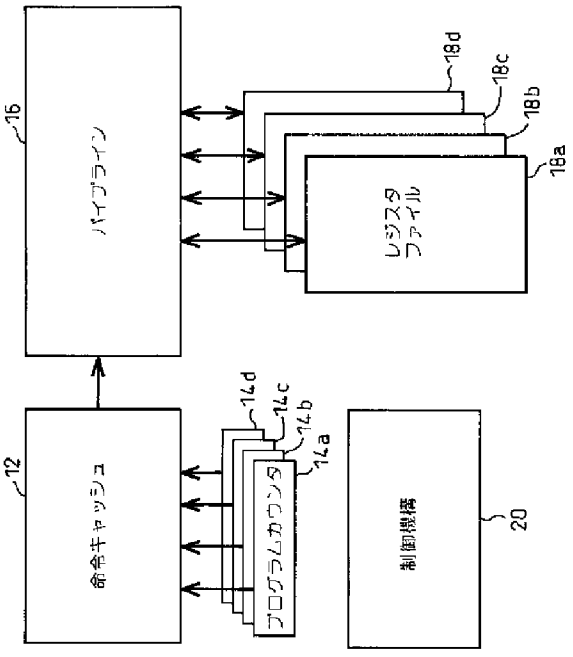
(71)出願人　000005223
富士通株式会社
神奈川県川崎市中原区上小田中４丁目１番１号
(72)発明者　安里　彰
神奈川県川崎市中原区上小田中４丁目１番１号　富士通株式会社内
(74)代理人　100077517
弁理士　石田　敬　（外４名）

Ｆターム(参考)　5B013 DD10
5B033 AA13 AA14 AA15 CA02
5B098 AA02 GA05 GC14

(54)【発明の名称】　マルチスレッド計算機

(57)【要約】
【課題】　並列化コンパイラで抽出された並列実行可能な複数プログラム流を単一のプロセッサでマルチスレッド実行させるマルチスレッド計算機を提供する。
【解決手段】　一定数のプログラム流を同時に処理するのに必要なプログラムカウンタ及びレジスタと、パイプライン制御を行う制御機構と、を備えるマルチスレッド計算機であって、該制御機構が、並列に実行可能なタスクの先頭アドレスをオペランドにて指定する第１の命令コードに応じて、新たなスレッドを生成して実行させる手段と、該並列に実行可能なタスクの終了を指定する第２の命令コードに応じて、該新たなスレッドを消滅させる手段と、を具備する。

図１



1

【特許請求の範囲】

【請求項１】　一定数のプログラム流を同時に処理するのに必要なプログラムカウンタ及びレジスタと、パイプライン制御を行う制御機構と、を備えるマルチスレッド計算機であって、該制御機構が、

並列に実行可能なタスクの先頭アドレスをオペランドにて指定する第１の命令コードに応じて、新たなスレッドを生成して実行させる手段と、

該並列に実行可能なタスクの終了を指定する第２の命令コードに応じて、該新たなスレッドを消滅させる手段と、

を具備するマルチスレッド計算機。

【請求項２】　一定数のプログラム流を同時に処理するのに必要なプログラムカウンタ及びレジスタと、パイプライン制御を行う制御機構と、を備えるマルチスレッド計算機であって、該制御機構が、

同一の先頭アドレスを有する並列に実行可能な複数のタスクの該先頭アドレスと該タスクの数とをオペランドにて指定する所定の命令コードに応じて、複数の新たなスレッドを生成して実行させる手段、

を具備するマルチスレッド計算機。

【請求項３】　前記制御機構は、生成されたスレッドにそれぞれ付加されるユニークなＩＤ番号を格納するスレッドＩＤレジスタを有し、該スレッドＩＤレジスタを指定する命令コードに応じて該命令コードが属するスレッドに対応するスレッドＩＤレジスタに格納された値を読み出す手段を更に具備する、請求項１又は請求項２に記載のマルチスレッド計算機。

【請求項４】　前記制御機構は、計算機の状況に応じて、命令コードに応じた新たなスレッドの生成を抑止して、シーケンシャルに命令を実行させるスレッド生成抑止手段を更に具備する、請求項１又は請求項２に記載のマルチスレッド計算機。

【請求項５】　前記制御機構は、生成された新たなスレッドのうち同時実行されるべきスレッドの数を制御するスレッド数制御手段を更に具備する、請求項１又は請求項２に記載のマルチスレッド計算機。

【発明の詳細な説明】

【０００１】

【発明の属する技術分野】本発明は、マルチスレッド計算機に関し、より詳細には、ハードウェアリソースの稼働効率の向上を図ったマルチスレッド計算機に関する。

【０００２】

【従来の技術】パイプライン計算機の実行スループットを向上させる手段として、マルチスレッド方式が提案され、実現されている。これは、パイプライン計算機が提供する並列の演算リソースを単一プログラム流が十分に使い切ることが一般に困難である点に注目し、複数のプログラム流をパイプラインに同時に投入することで、いわゆる空きスロットの割合を大幅に削減することを狙ったものである。

2

【０００３】一方、並列化コンパイラによってプログラムに存在する並列性を抽出し、マルチプロセッサで並列実行するアプローチが従来から行われている。

【０００４】

【発明が解決しようとする課題】ところで、一度に並列に処理することが可能な部分の大きさのことを並列性に関する粒度（granularity）と呼んでいるが、小さな粒度で並列処理しようとすると、オーバヘッドが大きくなり、あまり効率を向上させることができない。従来、マルチスレッド計算機は、小さな粒度での並列処理を目的とするものであり、プログラムの持つ大きな粒度の並列性には対応していない。その一方、並列化コンパイラによる並列性の抽出は、大きな粒度のレベルで行われ、ハードウェアリソースを必ずしもきめ細かいレベルで充分に活用しているとはいえない。

【０００５】本発明は、上述した問題点に鑑み、上述した二つの技術の融合を目指すものであり、その目的は、並列化コンパイラで抽出された並列実行可能な複数プログラム流を単一のプロセッサでマルチスレッド実行させるマルチスレッド計算機を提供することにある。

【０００６】

【課題を解決するための手段】上記目的を達成するために、本発明の第１の面によれば、一定数のプログラム流を同時に処理するのに必要なプログラムカウンタ及びレジスタと、パイプライン制御を行う制御機構と、を備えるマルチスレッド計算機であって、該制御機構が、並列に実行可能なタスクの先頭アドレスをオペランドにて指定する第１の命令コードに応じて、新たなスレッドを生成して実行させる手段と、該並列に実行可能なタスクの終了を指定する第２の命令コードに応じて、該新たなスレッドを消滅させる手段と、を具備するマルチスレッド計算機が提供される。

【０００７】また、本発明の第２の面によれば、一定数のプログラム流を同時に処理するのに必要なプログラムカウンタ及びレジスタと、パイプライン制御を行う制御機構と、を備えるマルチスレッド計算機であって、該制御機構が、同一の先頭アドレスを有する並列に実行可能な複数のタスクの該先頭アドレスと該タスクの数とをオペランドにて指定する所定の命令コードに応じて、複数の新たなスレッドを生成して実行させる手段、を具備するマルチスレッド計算機が提供される。

【０００８】また、本発明によれば、その制御機構は、生成されたスレッドにそれぞれ付加されるユニークなＩＤ番号を格納するスレッドＩＤレジスタを有し、該スレッドＩＤレジスタを指定する命令コードに応じて該命令コードが属するスレッドに対応するスレッドＩＤレジスタに格納された値を読み出す手段を更に具備する。

【０００９】また、本発明によれば、その制御機構は、計算機の状況に応じて、命令コードに応じた新たなスレ

ッドの生成を抑止して、シーケンシャルに命令を実行させるスレッド生成抑止手段を更に具備する。

【００１０】また、本発明によれば、その制御機構は、生成された新たなスレッドのうち同時実行されるべきスレッドの数を制御するスレッド数制御手段を更に具備する。

【００１１】

【発明の実施の形態】以下、添付図面を参照して本発明の実施形態について説明する。

【００１２】図１は、一般的なマルチスレッド計算機の動作を説明するためのブロック図である。同図において、１２は命令キャッシュ、１４ａ～１４ｄはそれぞれプログラムカウンタ、１６はパイプライン、１８ａ～１８ｄはそれぞれレジスタファイル、２０は制御機構である。プログラムカウンタおよびレジスタファイルは複数存在し、その個数はプロセッサ（計算機）がハードウェア的にサポートする同時実行可能なスレッド数の上限値である。

【００１３】実行中の各スレッドは、１４ａ～１４ｄ中の該当するプログラムカウンタの示すアドレスを用いて命令キャッシュ１２からフェッチされ、パイプライン１６に投入される。命令実行中にアクセスされるレジスタファイルは、１８ａ～１８ｄ中のいずれかから選択されて用いられる。このようなマルチスレッドプロセッサの構造は公知であり、本発明もそれを前提とするものである。

【００１４】図２は、並列化コンパイラによって抽出されたプログラムの構造の例を示す図である。同図の例では、当初、タスクＡのみが実行されており、その後、並列実行可能な複数のタスクＢ１～Ｂ４に制御が移り、最後は一本に合流してタスクＣとなる。

【００１５】本発明の一実施形態においては、図２のタスクＡの最後にthread_fork命令を置くことで、タスクＢ１～Ｂ４をマルチスレッド実行する枠組みが提供される。かかるthread_fork命令は、図３に示されるように、タスク開始アドレスをオペランドとして持つ。

【００１６】このthread_fork命令が実行されると、指定されたアドレスを先頭とするプログラム流が新たなスレッドとして定義され、公知の実行方式に従ってマルチスレッド実行される。図２に対応する図３の例では、並列化コンパイラによって、タスクＡの最後にタスクＢ１～Ｂ４に対応する４個のthread_fork命令が置かれることになる。

【００１７】また、スレッドの終わりを表すthread_terminate命令も用意され、この命令が実行されると公知の方式に従ってスレッドが消滅する。すなわち、並列化コンパイラによって、図４に示されるように、タスクＢ１の最後にはthread_terminate命令が置かれることになる。タスクＢ２～Ｂ４に関しても同様である。

【００１８】図２の構造の特殊な場合として、図５の例がある。図５はプログラムの流れとしては図２と同じであるが、プログラムがループ構造をとっており、タスクＢ１～Ｂ４が命令コード内の同一部分に位置する点が特殊である。本発明の一実施形態においては、この例に対応して、multi_thread_fork命令が用意される。

【００１９】このmulti_thread_fork命令は、スレッド開始アドレスと、生成されるべきスレッドの数とをオペランドとして持つ。すなわち、図５の例の場合、並列化コンパイラによって、図６に示されるように、タスクＡの命令列の最後にmulti_thread_fork命令が置かれ、そのオペランドにはタスクＢ１の開始アドレスとスレッド数４とが指定される。かくして、図５のタスクＢ１～Ｂ４をマルチスレッド実行させることが可能になる。

【００２０】なお、図５において、変数Ｉは、ループの繰り返し回数を意味するインデクス変数として用いられ、例えば、タスクＢ１で参照された場合には“１”の値、タスクＢ２で参照された場合には“２”の値となる。すなわち、これらのタスクが並列に実行される場合、同じ変数であっても異なる値が読み出される必要があり、何らかの対処が必要となる。本発明の一実施形態においては、スレッドＩＤレジスタによってこの問題が解決される。

【００２１】図７は、かかるスレッドＩＤレジスタがソースオペランドに指定された場合の動作を示すものである。同図において、２２はスレッド制御部、２４ａ～２４ｄはそれぞれスレッドＩＤレジスタ、２６は命令コード、２８はデコーダ、３０はセレクタであり、これらの要素は図１の制御機構２０に含まれる。

【００２２】スレッド制御部２２は、マルチスレッドプロセッサが一般的に有しているもので、並列実行中の複数のスレッドＩＤを格納するレジスタ群２４ａ～２４ｄを内部に備えている。

【００２３】図７に示されるように、デコーダ２８が出力する信号４０によってスレッドＩＤレジスタ群２４ａ～２４ｄの中から１個のレジスタが選択され、それに格納された値が読み出されて、常に信号４２として出力されている。この仕組みによってデコーダ２８がデコードしている命令のスレッドＩＤを信号４２の値として得ることが可能になっている。

【００２４】セレクタ３０は、複数のソースオペランド候補から、命令コードで指定される適切なものを選択するもので、その選択はセレクト信号４４の値で決定される。選択肢としては、通常のソースオペランド群４６に加えて信号４２で与えられるスレッドＩＤがあり、それらの中から選ばれた値が信号４８として出力される。

【００２５】スレッドＩＤレジスタをソースオペランドに持つ命令コード２６が与えられると、デコーダ２８がそれを解釈し、セレクタ３０が信号４２を選ぶような値をセレクト信号４４として設定する。これによって、命令が属するスレッドのＩＤをソースオペランド値として

読み出すことができる。

【００２６】このように、スレッドＩＤレジスタが命令のオペランドとして指定されると、各スレッドにユニークな番号（スレッドＩＤ）が読み出されるので、これを利用して、例えば、図８に示されるように、変数Ｉ（図５）にアクセスすれば、正しい値を参照することができる。なお、図８中、＜ｒ０＞はベースアドレスを与えるレジスタｒ０の内容、＜ＩＤ＞はスレッドＩＤレジスタの内容をそれぞれ示す。

【００２７】ところで、図２や図５のプログラムではコンパイラによって並列に実行可能なタスクが抽出されたが、本来はシーケンシャルなプログラムであり、並列実行しなくても正しく動作するものである。一方、マルチスレッドプロセッサは、動作状況によって最適な同時実行スレッド数が変化するのが一般的であり、システム全体の実効性能を高めるためには、スレッド数を適切に管理制御することが必須である。

【００２８】この点に鑑みて、本発明では、並列実行するスレッド数を制限すべく制御する枠組みが提供される。その一つは、thread_fork命令が実行されても、ハードウェア（すなわち制御機構２０）が自動的にプロセッサ（計算機）の稼動状況から、新たにスレッド生成することの得失を判断し、損失が大きいと判断した場合にはスレッド生成を行わないようにするものである。上記のようにプログラム自体は並列実行しなくても正しく動作するので、このような制御を行っても問題はない。

【００２９】また、もう一つは、この判断をハードウェアが自動的に行うのではなく、オペレーティングシステム等のソフトウェアが制御する手段を与えるものである。例えば、図９に示されるように、ソフトウェアによる設定が可能なスレッド生成抑止レジスタを用意し、オペレーティングシステムがスレッド生成を抑制すべきと判断した場合にこのレジスタをＯＮにセットすることで同様の効果を得ることができる。

【００３０】本発明によれば一般に複数のスレッドが同時に生成されることになる。図２や図５の例ではその数は４個である。しかし、前述のように、同時実行によって最大の効果が得られるスレッド数は、プログラムの性質やシステム全体の稼動状況に依存して変化する。そこで、同時に生成されたスレッド群のうち何個を同時実行すべきかを決定する手段を用意する。

【００３１】図１０は、ハードウェアが、状況に応じて、同時実行するスレッド数を自動的に決定し制御する機構を説明する図である。同図において、２２はスレッド制御部、１２は命令キャッシュ、５０は命令実行部、５２はモニタ部、５４はスレッド数指定レジスタである。

【００３２】命令キャッシュ１２の中には実行可能なスレッドの命令が複数存在し、その中からスレッド数指定レジスタ５４で指定された数のスレッドが選択されて並列実行される。モニタ部５２は、命令実行部５０の状況（演算リソースの使用状況など）を適宜モニタリングしながら、適切なスレッド数を決定し、信号５６を介してそのスレッド数をスレッド数指定レジスタ５４に設定する。

【００３３】図１１は、図１０におけるスレッド数指定レジスタ５４の設定をソフトウェアが実行することができるようにした構成を示す図である。図１１に示される構成は、図１０に示されるものに信号５８が加えられたものとなっている。コンパイラ等の解析により、適切な並列実行スレッド数が分かる場合には、図１０のようなハードウェア機能によらず、ソフトウェアによって信号５８を介してスレッド数指定レジスタ５４を設定する。

【００３４】以上、本発明を特にその好ましい実施の形態を参照して詳細に説明した。本発明の容易な理解のため、本発明の具体的な形態を以下に付記する。

【００３５】（付記１）　一定数のプログラム流を同時に処理するのに必要なプログラムカウンタ及びレジスタと、パイプライン制御を行う制御機構と、を備えるマルチスレッド計算機であって、該制御機構が、並列に実行可能なタスクの先頭アドレスをオペランドにて指定する第１の命令コードに応じて、新たなスレッドを生成して実行させる手段と、該並列に実行可能なタスクの終了を指定する第２の命令コードに応じて、該新たなスレッドを消滅させる手段と、を具備するマルチスレッド計算機。

【００３６】（付記２）　一定数のプログラム流を同時に処理するのに必要なプログラムカウンタ及びレジスタと、パイプライン制御を行う制御機構と、を備えるマルチスレッド計算機であって、該制御機構が、同一の先頭アドレスを有する並列に実行可能な複数のタスクの該先頭アドレスと該タスクの数とをオペランドにて指定する所定の命令コードに応じて、複数の新たなスレッドを生成して実行させる手段、を具備するマルチスレッド計算機。

【００３７】（付記３）　前記制御機構は、生成されたスレッドにそれぞれ付加されるユニークなＩＤ番号を格納するスレッドＩＤレジスタを有し、該スレッドＩＤレジスタを指定する命令コードに応じて該命令コードが属するスレッドに対応するスレッドＩＤレジスタに格納された値を読み出す手段を更に具備する、付記１又は付記２に記載のマルチスレッド計算機。

【００３８】（付記４）　前記制御機構は、計算機の状況に応じて、命令コードに応じた新たなスレッドの生成を抑止して、シーケンシャルに命令を実行させるスレッド生成抑止手段を更に具備する、付記１又は付記２に記載のマルチスレッド計算機。

【００３９】（付記５）　前記スレッド生成抑止手段は、自動的に計算機の状況を監視した結果として、シーケンシャルに命令を実行させるものである、付記４に記

載のマルチスレッド計算機。

【００４０】（付記６）　前記スレッド生成抑止手段は、計算機の状況を監視するソフトウェアの指示を受けて、シーケンシャルに命令を実行させるものである、付記４に記載のマルチスレッド計算機。

【００４１】（付記７）　前記制御機構は、生成された新たなスレッドのうち同時実行されるべきスレッドの数を制御するスレッド数制御手段を更に具備する、付記１又は付記２に記載のマルチスレッド計算機。

【００４２】（付記８）　前記スレッド数制御手段は、自動的に計算機の状況を監視した結果として、同時実行されるべきスレッドの数を制御するものである、付記７に記載のマルチスレッド計算機。

【００４３】（付記９）　前記スレッド数制御手段は、計算機の状況を監視するソフトウェアの指示を受けて、同時実行されるべきスレッドの数を制御するものである、付記７に記載のマルチスレッド計算機。

【００４４】

【発明の効果】以上説明したように、本発明によれば、並列化コンパイラで抽出された並列実行可能な複数プログラム流を単一のプロセッサでマルチスレッド実行させるマルチスレッド計算機が提供され、パイプライン制御を行うハードウェアリソースの稼働効率が更に向上せしめられる。

【図面の簡単な説明】

【図１】一般的なマルチスレッド計算機の動作を説明するためのブロック図である。

【図２】並列化コンパイラによって抽出されるプログラムの構造の例を示す図である。

【図３】命令列の終わりに配置されたthread_fork命令を示す図である。

＊【図４】命令列の終わりに配置されたthread_terminate命令を示す図である。

【図５】並列化コンパイラによって抽出されるプログラム構造の他の例を示す図である。

【図６】命令列の終わりに配置されたmulti_thread_fork命令を示す図である。

【図７】スレッドＩＤレジスタがソースオペランドに指定された場合の動作を示す図である。

【図８】インデクス変数へのアクセスを示す図である。

【図９】スレッド生成抑止レジスタについて説明するための図である。

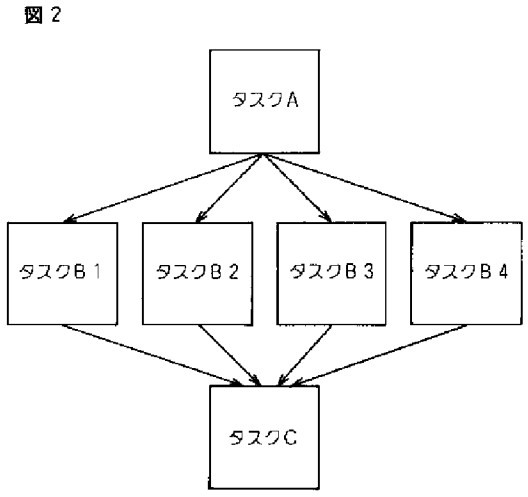
【図１０】ハードウェアが、状況に応じて、同時実行するスレッド数を自動的に決定し制御する機構を説明する図である。

【図１１】ソフトウェアが、状況に応じて、同時実行するスレッド数を自動的に決定し制御する機構を説明する図である。

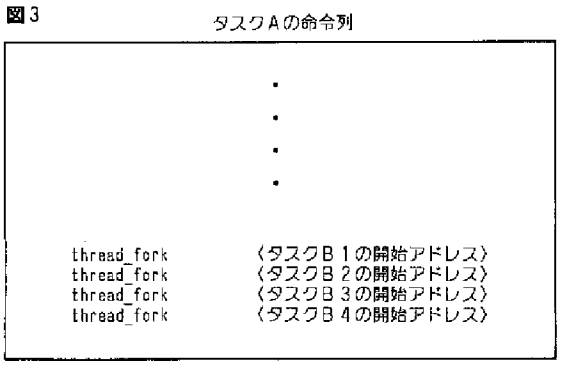
【符号の説明】

１２…命令キャッシュ
１４ａ～１４ｄ…プログラムカウンタ
１６…パイプライン
１８ａ～１８ｄ…レジスタファイル
２２…スレッド制御部
２４ａ～２４ｄ…スレッドＩＤレジスタ
２６…命令コード
２８…デコーダ
３０…セレクタ
５０…命令実行部
５２…モニタ部
５４…スレッド数指定レジスタ

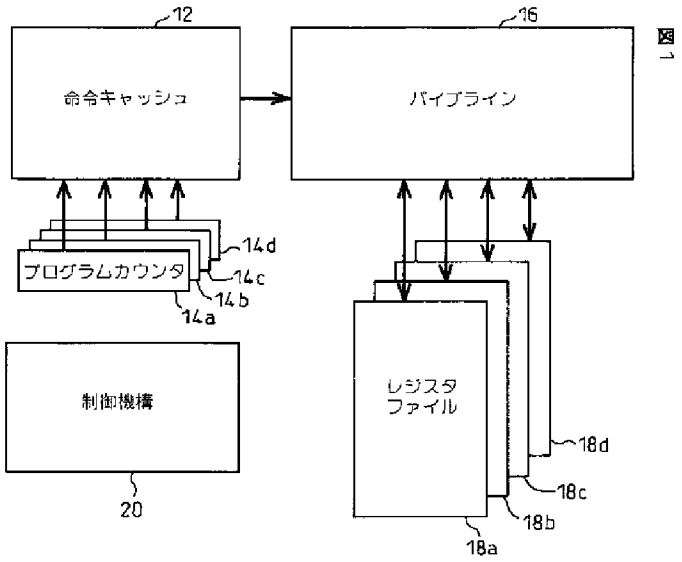
【図２】



【図３】



【図１】


12
16
命令キャッシュ
パイプライン
図１
14d
プログラムカウンタ
14c
14b
14a
制御機構
20
レジスタ
ファイル
18d
18c
18b
18a

【図４】

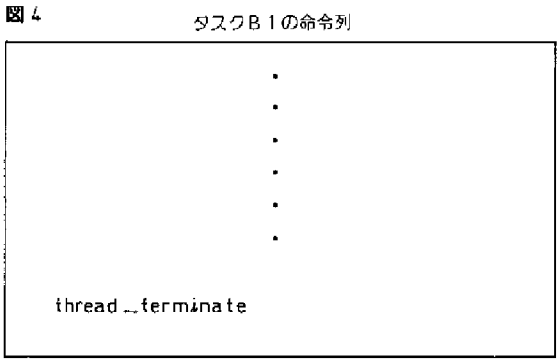


図４
タスクＢ１の命令列
thread_terminate

【図５】

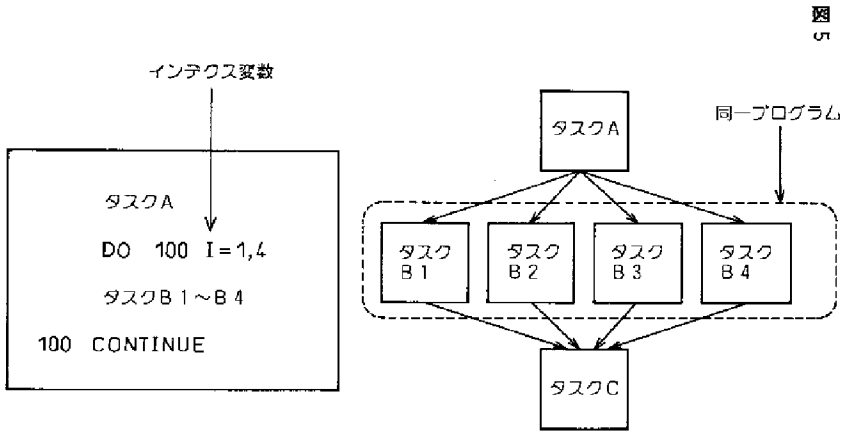
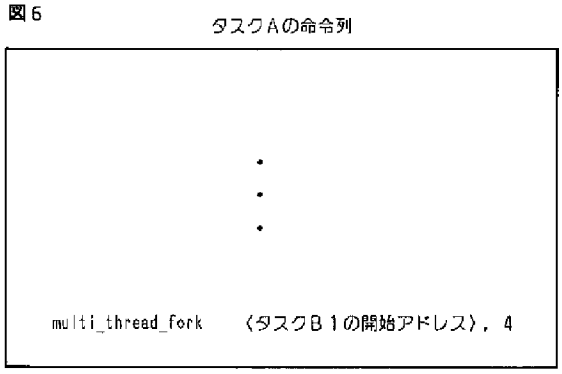


図５
インデクス変数
タスクA
DO 100 I=1,4
タスクＢ１～Ｂ４
100 CONTINUE
タスクA
同一プログラム
タスク
Ｂ１
タスク
Ｂ２
タスク
Ｂ３
タスク
Ｂ４
タスクC

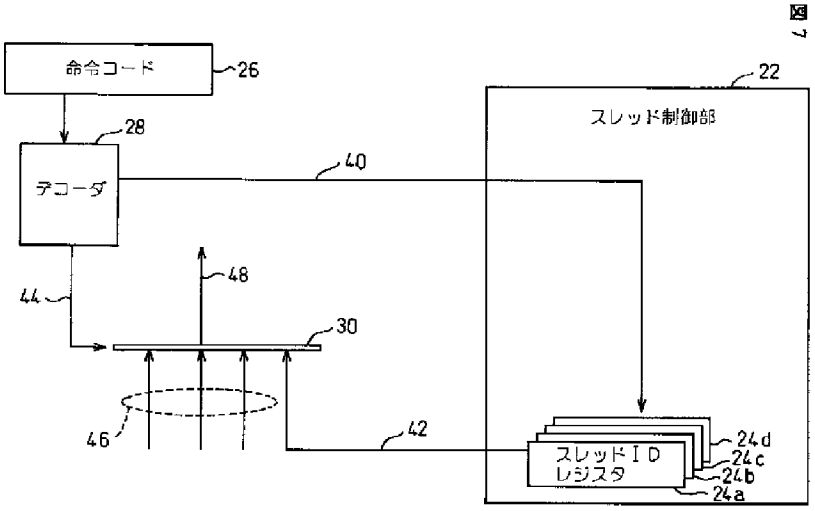
【図６】

図６



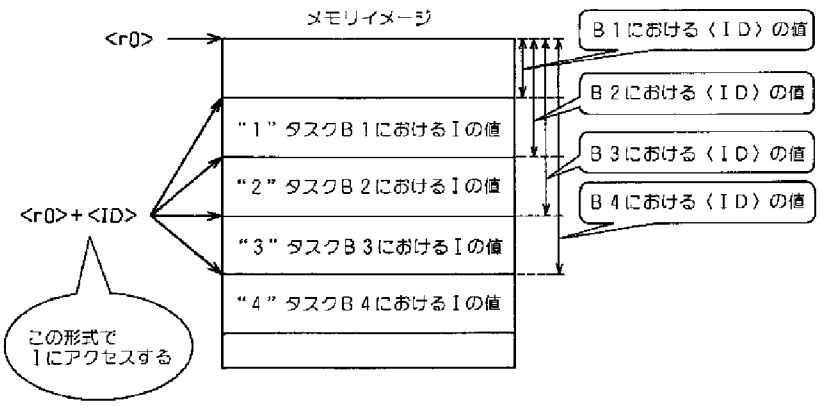
【図７】

図７



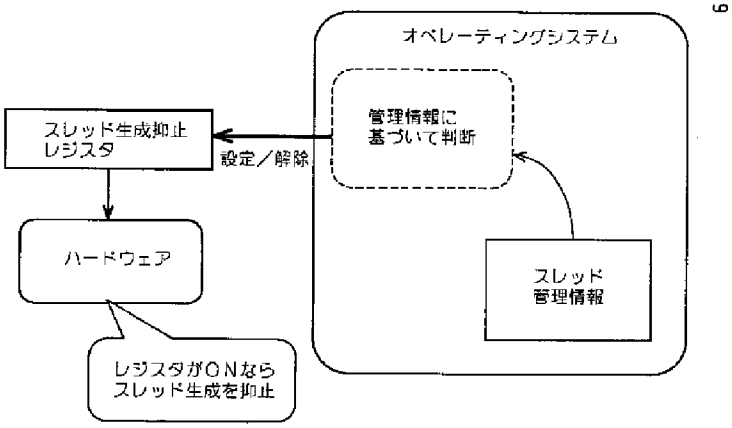
【図８】

図８



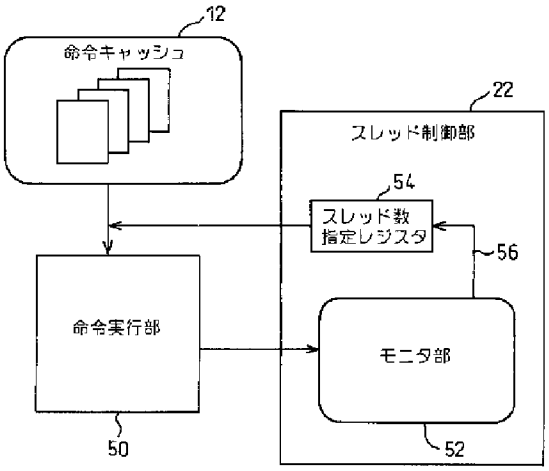
【図９】

図9



【図１０】

図10



【図１１】

図11